Developers Guide

Geo Web Solutions 2.1

Contents

1 Introduction

This is the Developers Guide for Geo Web Solutions 2.1. It describes the ways in which websites can be build with Geographical content using Geo Web Solutions components. A complete overview of XML requests that the Geo Web Solutions server supports is provided in this documentation. Also, the integration of the Flash client application in a website is explained, including the way the Flash object communicates with the web pages in which it is embedded.

1.1 Intended audience for this Developers Guide

This Developers Guide is primarily intended for website developers who need to build a site containing Geo Web Solutions components.

1.2 Document structure

This document starts with a brief overview of the application structure of Geo Web Solutions.

Note:
If you have developed websites based on previous releases of Geo Web Solutionsyou might want to check your code to determine if it still works with Geo Web Solutions version 2.1. Some XML requests have been slightly changed and/or expanded. Possible areas were you might want to check your code are:

- Expanded: The creation of XML requests to the FlexiMap.dll, see chapter 3.
- Expanded: The start parameters of the Flash object, see section 4.2.1.
- Changed: The capture of exposed parameters to JavaScript functions in webpage, see section 5.1.4.

2 Geo Web Solutions application structure

This chapter explains the program structure of Geo Web Solutions and the prescribed directory structure on the Geo Web Solutions server. A specific (virtual) directory structure must exist for Geo Web Solutions to function correctly. If you use the setup program the right directory structure will be created for you.

2.1 Webserver directory structure

The (IIS) webserver contains a directory named 'inetpub'. A subfolder of this directory named 'scripts' usually exists. This directory should contain the file **FlexiMap.dll.** This is the ISAPI program that forms the heart of the applications.
Directly under the root of the website a (virtual) directory must exist with the name **GeoWebSolutions**. This directory should contain (after a default installation) the following files and directories:

- **fmbase.swf:** the logic of the Flash Client application;
- **fmclient.swf:** the flash component to startup the application (this file can be stored with the websitescripts, but it is good practice to store this file in the same directory as the fmBase.swf);
- **fmGateway.swf**: flash component that is used to communicate between Javascript and the Flash application.
- **fmGateway.js:** javascript file with the available API functions for the Flash client.
- **security.xml:** xml file used by Flash to do a security check;
- **crossdomain.xml:** xml file that Flash uses to check which domains can have access to fmbase.swf (this file needs to be accessable at http://www.yoursite.com/GeoWebSolutions/crossdomain.xml) ;
- A subfolder named **assets** with (at least) the files containing the default icons used in the Flash Client:
    - iconsLarge_default.swf
    - iconsMap_default.swf
    - iconsSmall_default.swf
    - iconsLoader_default.swf

Defining the 'asset' files to use:
If you want to use your own iconfiles in a Geo Web Solutions "service" you can add them to this directory as long as you keep the naming convention in mind: *iconsLarge_myicons.swf, iconsMap_ myicons.swf, iconsSmall_myicons.swf* and *iconsLoader_myicons.swf*. All four should always be present for a specific iconset. The part of the filename after the "_" will be configured in the FlashGUI part of a service configuration:

```
  <Set name="window">
    <Element name="iconSet">myicons </Element>
 </Set>
```

This determines the icons used in the Flash Graphical User Interface when working with a specific service. The original fla files that were used to create the buttons that are used in Flash are part of the setup of GWS (section "samples and fla files for assets")

2.2 Registry entries for configuration and schema files

Geo Web Solutions uses a single XML configuration file that describes all the *services* that are being hosted by a specific Geo Web Solutions server. This file can be created and edited using the Geo Web Solutions Administrator. At startup the ISAPI program (Fleximap.dll) needs to be able to access and read this file. Information about the location of this configuration file is stored in a registry entry.

Geo Web Solutions uses XML schema files (.xsd) to validate all the incoming and outgoing xml requests. These schema files must be in a directory on the Geo Web Solutions server (default location c:\program files\GeoWebSolutions\meta). The path to this "meta" directory is also stored in a registry entry. You can find the following files in this directory:

- *FMConfig.xsd:* schema to parse the configuration file;
- *FMRequest.xsd:* schema for all the XML requests that Geo Web Solutions supports;
- *FMTypes.xsd:* schema for type definitions used in Geo Web Solutions;
- *FMPlugin_SaveRedline.xsd:* schema associated to the plugin functionality to save redline information via Geo Web Solutions (only needed if this functionality is available);

Registry keys:
In the *registry* two keys are used to indicate the location of these files:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Bentley\FlexiMap]
"LocationXML"="C:\Program Files\GeoWebSolutions\config\GeoWebSolutionsCfg.xml"
"SchemaPath"="C:\Program Files\GeoWebSolutions\meta"
```

Note that the setup program will create these registry entries for you.

3 Creating Geo Web Solutions XML Requests

3.1 Introduction

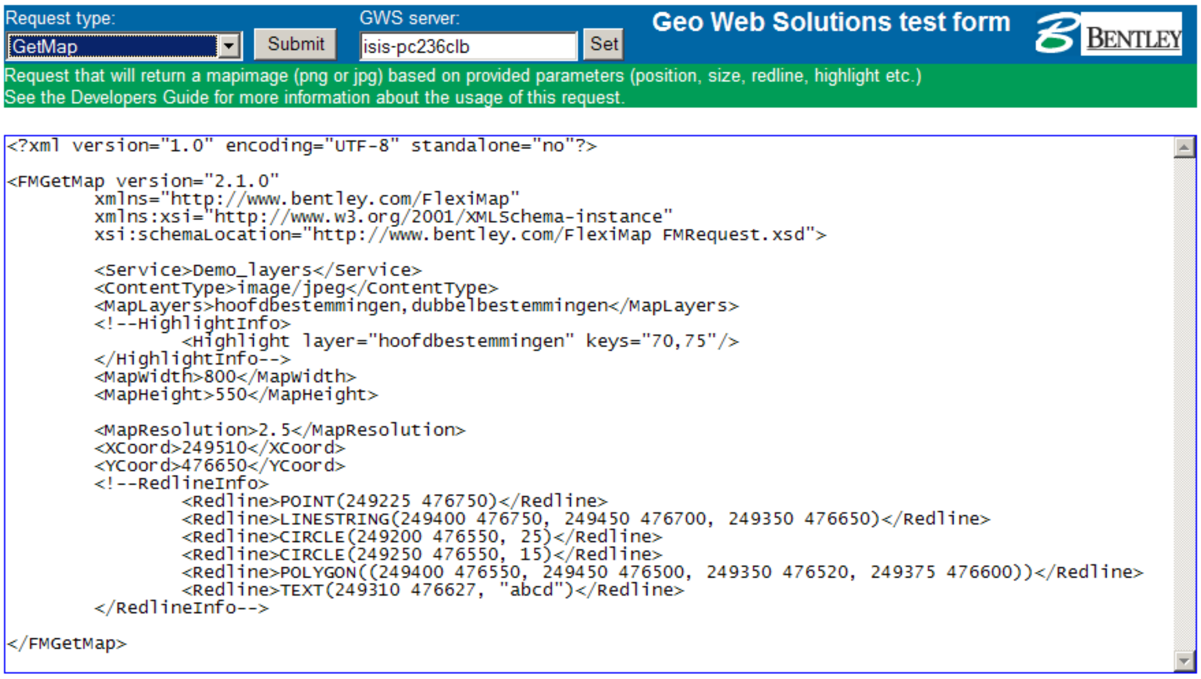
Geo Web Solutions enables developers to integrate geographical content in a website, or to create websites that present information obtained via “GIS” or “spatial” functionality offered by Geo Web Solutions. An example: if you want to to build a website that provides a list of *public services* or *restaurants* within a certain radius of an address. In your database you have a list of restaurants with their addresses and also coordinate-information for each address. Using Geo Web Solutions you can perform a spatial search that will respond with a set of attribute data based on the spatial parameters provided. This result can be shown -fully integrated- in the website either with or without a corresponding mapimage.
In order to support this kind of functionality Geo Web Solutions is build using a set of XML requests that each perform a specific task. In this chapter these XML requests are described and explained in detail. This will be done by giving an example for the usage of each request. The following requests are available for developers to integrate in a website:

Table 3.1: description of all supported requests:

Request:	*Description:*	*See section*
FMTest	Request to test if the Geo Web Solutions ISAPI dll is up and running.	3.2
FMGetServiceList	Returns a list with the names of all the available services on the addressed Geo Web Solutions server.	3.3
FMGetFlashGUI or FMGetCapabilities	Returns information about the configuration of the requested service. This request will be primarily used by the Geo Web Solutions Flash Client application, but can also be used to gather information about available reports and searches in a specific service.	3.4
FMGetServiceKey	Returns a valid service key that can be used in a GetMap, GetInfo, GetList, GetLocation, GetLocationMap and GetTooltip request if the requested service is secured with a service key.	3.5
FMGetExtent	Returns the maximum extent the map used in the requested service.	3.6
FMGetList	Returns a list of attribute values that meet the requirements of the input values (the “*questions*” in a defined search).	3.7
FMGetLocation	Returns X/Y-coordinates (centerpoint) and a MapResolution corresponding to the extent of the geometries found based on attribute values (MBR). This request uses as inputparameters attribute values that can be obtained using the FMGetList request.	3.7.1
FMGetMap	Returns a mapimage (in JPEG or PNG format) based on centerpoint X/Y-coordinates and MapResolution. This request can be expanded with additional information like redline and highlight information.	3.9
FMGetInfo	Returns a XML structured report containing all the attribute info linked to objects found by a SpatialMask or key/layer combination. A bufferdistance can be added to the request.	3.10
FMGetTooltip	Returns a XML structured report containing a “*tooltip*” value for the objects found at a specified coordinate (clickpoint).	3.11
FMGetReport	Combination of a FMGetList and a FMGetInfo request. Returns a XML structured report containing all the attribute info linked to objects found with the provided attribute (search) values.	3.12

FMGetReportByKeyInfo	Returns a XML structured report containing attribute info based on (graphical) keys/layername.	3.13
FMGetReportByKeys	Returns a XML structured report containing attribute info based on keys. A reportname must always be specified.	3.14
FMGetLocationByKeyInfo	Returns X/Y-coordinates (centerpoint) and a MapResolution corresponding to the extent of the geometries found based on key information (MBR). This request uses as key(s) and a layername as input parameters (the key/layer information is always available for each record in the response to a FMGetInfo request).	3.15
FMGetMapByKeyInfo	Combination of a FMGetMap and a FMGetLocationByKeyInfo request that returns a mapimage (JPEG or PNG) based on key/layer information.	3.16
FMGetLocationMap	Combination of a FMGetMap and a FMGetLocation request that returns a mapimage (JPEG or PNG) based on attribute information. **This is the only request that can be executed using either XML or a URL with parameters.**	3.17

Fig: Impression of the testform for testing all possible xml requests



Request that will return a mapimage (png or jpg) based on provided parameters (position, size, redline, highlight etc.)
See the Developers Guide for more information about the usage of this request.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<FMGetMap version="2.1.0"
        xmlns="http://www.bentley.com/FlexiMap"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.bentley.com/FlexiMap FMRequest.xsd">

        <Service>Demo_layers</Service>
        <ContentType>image/jpeg</ContentType>
        <MapLayers>hoofdbestemmingen,dubbelbestemmingen</MapLayers>
        <!--HighlightInfo>
                <Highlight layer="hoofdbestemmingen" keys="70,75"/>
        </HighlightInfo-->
        <MapWidth>800</MapWidth>
        <MapHeight>550</MapHeight>

        <MapResolution>2.5</MapResolution>
        <XCoord>249510</XCoord>
        <YCoord>476650</YCoord>
        <!--RedlineInfo>
                <Redline>POINT(249225 476750)</Redline>
                <Redline>LINESTRING(249400 476750, 249450 476700, 249350 476650)</Redline>
                <Redline>CIRCLE(249200 476550, 25)</Redline>
                <Redline>CIRCLE(249250 476550, 15)</Redline>
                <Redline>POLYGON((249400 476550, 249450 476500, 249350 476520, 249375 476600))</Redline>
                <Redline>TEXT(249310 476627, "abcd")</Redline>
        </RedlineInfo-->

</FMGetMap>
```

The next paragraphs will explain each request in detail. The parts of a request in red are optional parameters.

3.2 The FMTest request

The FMTest request is a simple request to determine if the Geo Web Solutions ISAPI dll is “Up and Running”. This request can be useful at installation time, or in a debugging process. The FMTest request needs to have the following structure:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<FMTest
       version="2.1.0"
       xmlns="http://www.bentley.com/FlexiMap"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.bentley.com/FlexiMap FMRequest.xsd" />
```

The XML response to this request has the following structure:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<FMSendTest xmlns="http://www.bentley.com/FlexiMap" version="2.1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.bentley.com/FlexiMap FMResponse.xsd">
  Geo Web Solutions is up and running
</FMSendTest>
```

If the returned response is different from the one shown here, something is wrong. If the configuration file cannot be parsed by the Geo Web Solutions ISAPI dll a XML exception will be generated. If you receive the message "*The specified module could not be found*" the ISAPI dll cannot be loaded. In this case you should check if it is present at the right location and if all it's dependencies are installed on the server.

Note:
The setup will install dependencies in a subfolder named *bin*. It will also expand the system environment variable *path* to include this directory. Before you start using Geo Web Solutions you need to reboot the computer. If you forget this the path variable is not re-initialized and Geo Web Solutions will respond with the "*The specified module could not be found*" error.

3.3 The FMGetServiceList request

The FMGetServiceList request is used to get a list of available services from a specific Geo Web Solutions server. The XML response will contain a list of available services.
All the other requests require this service name. For example, once a service name is known you can execute a FMGetFlashGUI request to get detailed information about the available reports and searches in this service. A FMGetServiceList request needs to have the following structure:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<FMGetServiceList
  version="2.1.0"
  xmlns="http://www.bentley.com/FlexiMap"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bentley.com/FlexiMap FMRequest.xsd"/>
```

The XML response to this request has the following structure:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<FMSendServiceList
  xmlns="http://www.bentley.com/FlexiMap"
  version="2.1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bentley.com/FlexiMap FMSendServiceList.xsd">

  <Services>
    <Service>demo</Service>
    <Service>demo_layers</Service>
  </Services>

</FMSendServiceList>
```

The names returned in the FMSendServiceList will be used in all other XML requests that can be processed by the Geo Web Solutions server.

3.4 The FMGetFlashGUI request (also named FMGetCapabilities)

Once you have the name of the service that you want to use, you can execute a FMGetFlashGUI or FMGetCapabilities request to get information about this specific service configuration. This request is mainly used by the Flash Client application that is delivered with the product. The response to this

request will provide flash the information it needs to initialize itself. However, also information is provided that can be useful if you want to use the service in a non flash environment. You might want to know what type of (attribute) information is provided by a service (available reports), what searches can be executed and if the service is *secured* with a service key).
A FMGetFlashGUI/FMGetCapabilities request has the following structure:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<FMGetFlashGUI
  version="2.1.0"
  xmlns="http://www.bentley.com/FlexiMap"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bentley.com/FlexiMap FMRequest.xsd">

  <Service>demo</Service>
  <MapWidth>400</MapWidth>
  <MapHeight>550</MapHeight>

</FMGetFlashGUI>
```

The XML response to this request has the following structure:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<FMSendFlashGUI
  xmlns="http://www.bentley.com/FlexiMap"
  version="2.1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bentley.com/FlexiMap FMSendFlashGUI.xsd">

  <Service>demo</Service>

  <GeneralProperties>
    <ZoomFactor>2.000000</ZoomFactor>
    <UseJavascript>true</UseJavascript>
    <UseServiceKey>false</UseServiceKey>
    <UseTooltips>true</UseTooltips>
    <MinResolution>0.100000</MinResolution>
    <MaxResolution>5.000000</MaxResolution>
    <MapResolution>2.636364</MapResolution>
    <XCoord>249500.000000</XCoord>
    <YCoord>476675.000000</YCoord>
  </GeneralProperties>
  <Interface>
    [part that defines texts in the Flash interface]
  </Interface>
  <Tools>
    [part that defines available tools in the Flash interface]
  </Tools>
  <RedlineOptions>
    [part that defines redline options in the Flash interface]
  </RedlineOptions>
  <SpatialQueryOptions>
    [part that defines spatial query options the Flash interface]
  </SpatialQueryOptions>
  <ZoomSteps>
    [part that defines available zoomsteps in the Flash interface]
  </ZoomSteps>
  <LegendLayers>
    [defines available maplayers and the hierarchical presentation in the Flash interface]
  </LegendLayers>
  <Reports>
    <Report alias=”adresses”>adres</Report>
    <Report alias=”building”>bouwvlak</Report>
    <Report alias=”parcels”>kadaster</Report>
  </Reports>
  <SearchQueries>
    <SearchQuery alias=”” name="adres">
      <Columns>
        <Column label="Straatnaam" type="alphanumeric">STRAAT</Column>
        <Column label="Huisnummer" type="numeric">HUISNR</Column>
      </Columns>
    </SearchQuery>
    <SearchQuery alias=”zoning plan” name="bplan">
      <Columns>
        <Column label="Plannaam" type="alphanumeric">Omschrijving</Column>
      </Columns>
    </SearchQuery>
  </SearchQueries>
</FMSendFlashGUI>
```

Note: this request can also be named FMGetCapabilities. It returns exactly the same information as shown here. It is a synonym.

3.5 The FMGetServiceKey request

The FMGetServiceKey request only needs to be used if a service is secured with a service key. To determine if a service is secured you can execute a FMGetFlashGUI request that will respond with *<UseServiceKey>true</UseServiceKey>* if the service is secured. If this is the case the following requests will only work if a valid service key is provided:

- FMGetMap;
- FMGetList;
- FMGetLocation;
- FMGetInfo;
- FMGetTooltip;
- FMGetReport;
- FMGetReportByKeyInfo;
- FMGetReportByKeys;
- FMGetLocationByKeyInfo;
- FMGetMapByKeyInfo.

To obtain a valid service key, execute the FMGetServiceKey request from a serverside script on a computer with an IP-address known to the Geo Web Solutions configuration. Only computers with a IP-address that is configured in Geo Web Solutions (defined in the <ServiceKeyHosts> element in the configuration XML) are allowed to successfully request a service key.
The XML response will contain a service key that will be valid for a defined period of time (each service can have a different expiration time for a key). This service key must be present in all the above mentioned requests.

The FMGetServiceKey request has the following structure:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<FMGetServiceKey
  version="2.1.0"
  xmlns="http://www.bentley.com/FlexiMap"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bentley.com/FlexiMap FMRequest.xsd">

  <Service>demo</Service>

</FMGetServiceKey>
```

The XML response to this request has the following structure:

If FMGetServiceKey is requested for a service that is NOT secured:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<FMServiceExceptionReport xmlns="http://www.bentley.com/FlexiMap" version="2.1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.bentley.com/FlexiMap FMServiceException.xsd">

 <ServiceException code="0x8243F001">
   <Reason>Service key exception</Reason>
   <Description>The service 'demo' doesn't require a service key</Description>
 </ServiceException>

</FMServiceExceptionReport>
```

If FMGetServiceKey is requested for a service that is secured:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<FMSendServiceKey
  xmlns="http:// www.bentley.com /FlexiMap"
  version="2.1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bentley.com/FlexiMap FMSendServiceKey.xsd">

  <Service>demo</Service>
  <Key>7981507462956756273</Key>

</FMSendServiceKey>
```

The returned key needs to be stored somewhere and will be used in all the requests that require a service key.

Note: In the requests that can be secured with a servicekey the position of the **<ServiceKey>** element is always directly after the **<Service>** element.

3.6 The FMGetExtent request

The FMGetExtent request is a simple request that will return the maximum extent of the mapimage that can be generated with the requested service. This maximum extent is defined by the x- and y coordinates of the centerpoint and the MapResolution corresponding to the full extent (MapResolution is the number of coordinate units per pixel).

The FMGetExtent request has the following structure:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<FMGetExtent
  version="2.1.0"
  xmlns="http://www.bentley.com/FlexiMap"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bentley.com/FlexiMap FMRequest.xsd">

  <Service>demo</Service>
  <MapWidth>400</MapWidth>
  <MapHeight>550</MapHeight>

</FMGetExtent>
```

The XML response to this request has the following structure:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<FMSendExtent
  xmlns="http://www.bentley.com/FlexiMap"
  version="2.1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bentley.com/FlexiMap FMSendExtent.xsd">

  <Service>demo</Service>
  <MapResolution>12.636364</MapResolution>
  <XCoord>249500.000000</XCoord>
  <YCoord>476675.000000</YCoord>

</FMSendExtent>
```

3.7 The FMGetList request

The FMGetList request can be used to execute attribute searches. A service can be defined with multiple attribute searches. The FMGetFlashGUI will have provided you with information about available searches. The response from a FMGetList request is a list of attribute values that correspond with the input provided.
You can use the FMGetList request for example to present select boxes to the users in your website (see provided Test page for Flash Client). The results of an attribute search like this can be used to start the Geo Web Solutions Flash application with the correct parameters (see next chapter) or to get a mapimage in (raster format) using the FMGetLocation or FMGetMap request.

The FMGetList request needs to have the following structure:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<FMGetList
  version="2.1.0"
  xmlns="http://www.bentley.com/FlexiMap"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bentley.com/FlexiMap FMRequest.xsd">

  <Service>demo</Service>
  <ServiceKey>797250026230983 4033</ServiceKey> <!-- optional element -->
  <SearchQuery name="Address">
    <Column name="STREET">Mainstreet</Column>
    <Column name="NUMBER"></Column>
  </SearchQuery>

</FMGetList>
```

An attribute search can contain one or multiple questions.

The way the <Column> element is used in the request defines the returned result:

Using a search named “Address” with two questions defined (streetname and housenumber), the following syntax applies if you want to:

- **Get a list of all unique Streetnames** (first question of this search) → enter a <Column> element with the name of the database column associated to the first question. Do not enter any values for this question[1] :
  <SearchQuery name="Address">
  <Column name="STREET"></Column>
  </SearchQuery>
- **Get a list of all Streetnames that start with “Pla”** → enter a <Column> element with the name of the database column associated to the first question. Enter the three characters for this question:
  <SearchQuery name="Address">
  <Column name="STREET">pla</Column>
  </SearchQuery>
- **Get a list of all valid housenumbers** (second question of attribute search) **that can be found for “Plaza de España”** → Enter a <Column> element with the name and value of the street and an empty <Column> element for the second question:
  <SearchQuery name="Address">
  <Column name="STREET">Plaza de Catalunya</Column>
  <Column name="NUMBER"></Column>
  </SearchQuery>
- Et cetera.

The XML response to the FMGetList request has the following structure:

[1] A question can be configured to always need at least some characters to be entered in order to start the attribute search. If this is the case then you cannot leave the <Column> element empty.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<FMSendList
  xmlns="http://www.bentley.com/FlexiMap"
  version="2.1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bentley.com/FlexiMap FMSendList.xsd">

  <Service>demo</Service>
    <Column>
      <Item>2</Item>
      <Item>3</Item>
      <Item>4</Item>
      <Item>5</Item>
   </Column>

</FMSendList>
```

3.7.1 Using operators in a FMGetList request

From version 2.1 onward it is possible to use operators for each searchcolumn. It is also possible to use the same column more then once in a FMGetList request. This allows you to use this request to create “Query builder” functionality in your website.

The following operators are available for each column:

Operator	Description	Remarks
No operator or “=”	Equals	Default value if no operator is used
“<>” or “!=”	Not equals	
“<”	Smaller then	< must be escaped in XML → &lt;
“>”	Larger then	> must be escaped in XML → &gt;
“<=”	Smaller then or equals	
“>=”	Larger then or equals	
“in”	Value in the comma separated list must occurr in the database	
“not in”	Value in the comma separated list must NOT occurr in the database	
“like”	Value must be similar to.. (see using wildcards)	
“not like”	Value must NOT be similar to.. (see using wildcards)	

Note: in XML you need to “escape” some characters by using character entities. “<” becomes “&lt;” and “>” becomes “&gt;”.

Using wildcards

The “like” and “not like” operators can be used to support searching with wildcards. It works similar to a database query in Oracle and MS-databases (SQL-Server, Access). When executing the search the MS-syntax will be transormed to the Oracle-syntax, because this syntax works in both environments.
Supported are:

- “_”, “?”: single character;
- “%”, “*”: multiple characters.

Here are some examples of using operators in a FMGetList request:

Request: give me all the housenumbers that you can find in the Bantingstraat that fall between 12 and 16, counting 16.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<FMGetList
        version="2.1.0"
        xmlns="http://www.bentley.com/FlexiMap"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.bentley.com/FlexiMap FMRequest.xsd">

        <Service>demo_Layers</Service>
        <SearchQuery name="Address">
                <Column name="STRAAT">Bantingstraat</Column>
                <Column name="HUISNR" operator="&gt;">12</Column>
                <Column name="HUISNR" operator="&lt;=">16</Column>
        </SearchQuery>

</FMGetList>
```

Returns:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<FMSendList xmlns="http://www.bentley.com/FlexiMap" version="2.1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.bentley.com/FlexiMap FMResponse.xsd">
 <Service>demo_layers</Service>
 <Column>
  <Item>13</Item>
  <Item>14</Item>
  <Item>15</Item>
  <Item>16</Item>
  </Column>
</FMSendList>
```

Request: give me all the streetnames that have the characters “an” at second and third postion.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<FMGetList
        version="2.1.0"
        xmlns="http://www.bentley.com/FlexiMap"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.bentley.com/FlexiMap FMRequest.xsd">

        <Service>demo_Layers</Service>

        <SearchQuery name="Address">
                <Column name="STRAAT" operator=”like”>?an*</Column>
        </SearchQuery>

</FMGetList>
```

Returns:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<FMSendList xmlns="http://www.bentley.com/FlexiMap" version="2.1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.bentley.com/FlexiMap FMResponse.xsd">
 <Service>demo_layers</Service>
 <Column>
  <Item>BANTINGSTRAAT</Item>
  <Item>LANDSTEINERSTRAAT</Item>
  </Column>
</FMSendList>
```

3.8 The FMGetLocation request

After performing (a number of) FMGetList requests you will have all the attribute values that you need as input for a FMGetLocation request. This request is designed to provide you with information about the geographical location of objects that meet the requirements of the attribute values. The X/Y-coordinate information of the centerpoint and the MapResolution returned can be used to create a FMGetMap or FMGetInfo request.

The FMGetLocation request has the following structure:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<FMGetLocation
  version="2.1.0"
  xmlns="http://www.bentley.com/FlexiMap"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bentley.com/FlexiMap FMRequest.xsd">

  <Service>demo</Service>
  <ServiceKey>7972500262309834033</ServiceKey> <!-- optional element -->
  <SearchQuery name="Adres">
    <Column name="STREET">MAINSTREET</Column>
    <Column name="NUMBER">16</Column>
  </SearchQuery>
  <MapWidth>600</MapWidth>
  <MapHeight>550</MapHeight>

</FMGetLocation>
```

The XML response to the FMGetLocation request has the following structure:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<FMSendLocation
  xmlns="http://www.bentley.com/FlexiMap"
  version="2.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bentley.com/FlexiMap FMSendLocation.xsd">

  <Service>demo</Service>
  <MapResolution>0.500000</MapResolution>
  <XCoord>249301.100000</XCoord>
  <YCoord>476669.500000</YCoord>

</FMSendLocation>
```

The FMGetLocation request does not require that ALL questions for as search are answered. In the above example the answer to the second question `<Column name="NUMBER">16</Column>` can be ommitted. The result would then be a centerpoint/mapresolution combination reflecting the MBR off all the adresses found in the specified street.

3.8.1 Using operators in a FMGetLocation request

See paragraph 3.7.1. The operators explained there also apply to the FMGetLocation request.

3.9 The FMGetMap request

One of the most important requests is probably the FMGetMap request. It is the only request that does not return a XML response. The result of an FMGetMap request is a mapimage in rasterformat. The two supported file types are:

PNG	<ContentType>image/png</ContentType>
JPEG	<ContentType>image/jpeg</ContentType>

Based on the X/Y-coordinates of the centerpoint of the mapimage and a mapresolution (this information can be provided by a FMGetLocation request) and a mapwidth and mapheight (in pixels) a mapimage can be returned. The (graphic) information that will be shown in this image is defined in the service.

Defining Maplayers to include:

If you are requesting mapimages from a service that uses a WMS mapserver to deliver these mapimages, the possibility exists to request specific maplayers. These maplayers need to be defined in the service configuration (see the Geo Web Solutions Administration Guide).
Adding the following element to the request <MapLayers>Zoningplan,Parcels</MapLayers> will generate a mapimage with only the defined layers visible. If this element is not present in the request a set of default layers will be shown.

Highlighting geometries based on identifiers (keys):

A FMGetMap request can be expanded to highlight geometries based on layer/key combinations. The following elements need to present in the request:
<HighlightInfo>

```
  <Highlight layer="Parcels" keys="70,75"/>
  <Highlight layer="ZoningPlan" keys="134"/>
</HighlightInfo>
```

The **layer** attribute needs to be present, because Geo Web Solutions needs to know where to search for coordinate information. The **keys** attribute defines which geometries in the layer need to be highlighted. You can highlight geometries in more then one layer at once. The color used to highlight geometries is defined in the service configuration.

Note: Key and layer information will always be provided in the response to a FMGetInfo request. For each key found the layer in which it has been found will be part of the response.

Redline information:

A FMGetMap request can also be expanded with "redline" information. If coordinate information is provided in a request, the Geo Web Solutions server will return a map image with the redline information "burned" in the map. Color, linewidth etc. used for redlining objects on the map is defined in the service configuration for each type of redline object available. The following redline objects are available:

POINT:	`<Redline>POINT(249225 476750)</Redline>`
LINESTRING:	`<Redline>LINESTRING(249400 476750, 249450 476700, 249350 476650)</Redline>`
CIRCLE:	`<Redline>CIRCLE(249200 476550, 25)</Redline>`
POLYGON:	`<Redline>POLYGON((249400 476550, 249450 476500, 249350 476520, 249375 476600))</Redline>`
TEXT:	`<Redline>TEXT(249310 476627, "abcd")</Redline>`

An example of a FMGetMap XML request is:

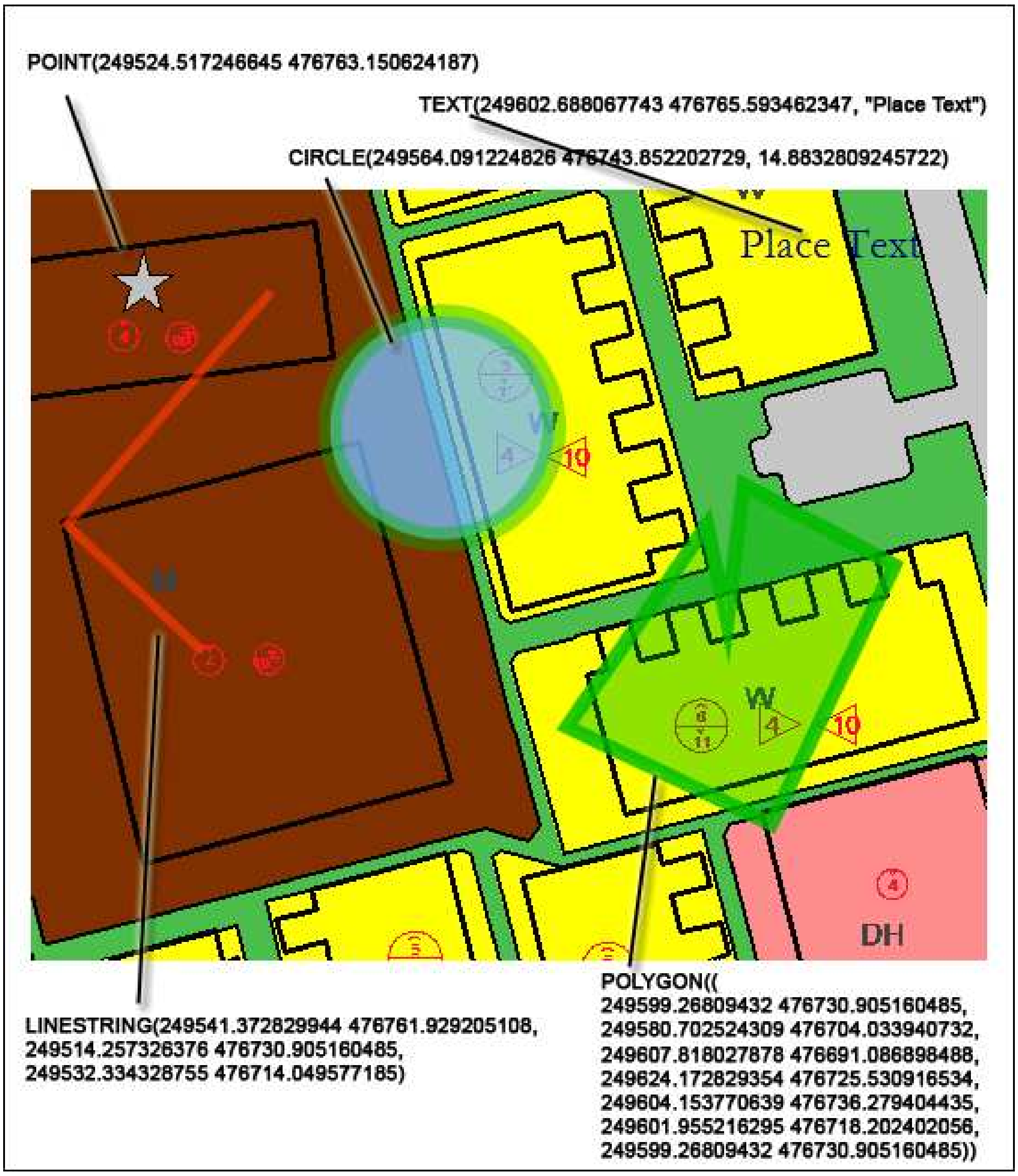
```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<FMGetMap
  version="2.1.0"
  xmlns="http://www.bentley.com/FlexiMap"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bentley.com/FlexiMap FMRequest.xsd">

  <Service>demo</Service>
  <ServiceKey>7972500262309834033</ServiceKey> <!-- optional element -->
  <ContentType>image/png</ContentType>
  <MapLayers>Zoningplan,Parcels</MapLayers>
  <HighlightInfo>
    <Highlight layer="Parcels" keys="70,75"/>
  </HighlightInfo>
  <MapWidth>520</MapWidth>
  <MapHeight>422</MapHeight>
  <MapResolution>0.244283815931462</MapResolution>
  <XCoord>249573.129726015</XCoord>
  <YCoord>476724.065213639</YCoord>

  <RedlineInfo>
    <Redline>POINT(249524.517246645 476763.150624187)</Redline>
    <Redline>LINESTRING(249541.372829944 476761.929205108,249514.257326376
476730.905160485,249532.334328755 476714.049577185)</Redline>
    <Redline>CIRCLE(249564.091224826 476743.852202729, 14.8832809245722)</Redline>
    <Redline>POLYGON((249599.26809432 476730.905160485,249580.702524309
476704.033940732,249607.818027878 476691.086898488,249624.172829354
476725.530916534,249604.153770639 476736.279404435,249601.955216295
476718.202402056,249599.26809432 476730.905160485))</Redline>
    <Redline>TEXT(249602.688067743 476765.593462347, "Place Text")</Redline>
  </RedlineInfo>

</FMGetMap>
```

The image shown below is the result of a FMGetMap request. It is a mapimage with all possible redline options available.



```
POINT(249524.517246645 476763.150624187)

TEXT(249602.688067743 476765.593462347, "Place Text")

CIRCLE(249564.091224826 476743.852202729, 14.6632809245722)

LINESTRING(249541.372829944 476761.929205108,
249514.257326376 476730.905160485,
249532.334328755 476714.049577185)

POLYGON((
249599.26809432 476730.905160485,
249580.702524309 476704.033940732,
249607.818027878 476691.086898488,
249624.172829354 476725.530916534,
249604.153770639 476736.279404435,
249601.955216295 476718.202402056,
249599.26809432 476730.905160485))
```

3.10 The FMGetInfo request

The FMGetInfo request is the one you need if you want to get attribute reports based on location information. A FMGetInfo request will always need coordinate information that determines the location or shape that you want to retrieve informaion about.

In the service configuration one or more reports can be linked to a layer. If you do not specify one of the following optional elements:
<ReportName>
<MapLayers>
a FMGetInfo request will respond with the attribute information it can find at the given location for all the reports defined in the service. Of course only attribute reports will be generated if geometries can be found at the supplied location.

However, it is also possible to create a FMGetInfo request that only returns a specified report by adding the <ReportName> element with the name of the report that you are interested in. The available report names are listed in the response to the FMGetFlashGUI request.

By default, a FMGetInfo request will start looking for geometries in all the layers definied in the service configuration. However, by adding the <MapLayers> element you can specify which layers need to be searched.

There are two ways to define the location/shape that you use as an input parameter in a FMGetInfo request.

1. If you only have a clickpoint then you can use the <ClickPoint> element with coordinate (X and Y) information.
2. If you want to retrieve attribute information based on point, line, circle or polygon information you can also use the <SpatialMask> element. If this method is used, the syntax of the coordinate information is the same as the redline information in an FMGetMap request.

Here is an example of an FMGetInfo request:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<FMGetInfo
  version="2.1.0"
  xmlns="http://www.bentley.com/FlexiMap"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bentley.com/FlexiMap FMRequest.xsd">

  <Service>serv1</Service>
  <ServiceKey>797250026230983403</ServiceKey> <!-- optional element -->
  <ClickPoint>249323 476652</ClickPoint>
[or]
  <SpatialMask>POINT(249323 476652)</SpatialMask>
[or]
  <SpatialMask>LINESTRING(249541 476761,249532. 476714)</SpatialMask>
[or]
  <SpatialMask>CIRCLE(249564 476743, 14.2)</SpatialMask>
[or]
  <SpatialMask>POLYGON((249599 476730,249580 476704,249607 476691,24960 476718,249599
476730))</SpatialMask>
  <BufferDist>25</BufferDist><!-- optional element for specifying a buffer distance -->
  <MapLayers>Zoningplan,Parcels</MapLayers> <!-- optional element -->
  <MapResolution>0.5</MapResolution>
  <MapWidth>600</MapWidth>
  <MapHeight>500</MapHeight>
  <ReportName>name of report </ReportName> <!-- optional element -->

</FMGetInfo>
```

The result of the FMGetInfo request will have a structure as shown below. It always starts with the clickpoint or spatialmask that was used in the request. Then for each Report:

- **header information**: name of each databasecolumn and alias if specified.
- A collection of **records** containing 1 or n records.
- Each record contains
  - the key of the geometry found (i.e. to be used to highlight geometries)
  - the layer in which it was found (i.e. to be used to highlight geometries)
  - values for all the databasecolumns defined in the header information

The response always ends with the *X/Y-coordinates* of the centerpoint and a *mapresolution* corresponding with the extent(s) of **all** the geographical object(s) found (MBR).

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<FMSendInfo xmlns="http://www.bentley.com/FlexiMap" version="2.1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.bentley.com/FlexiMap FMResponse.xsd">
  <Service>demo</Service>
  <SpatialMask>CIRCLE(249571.104472 476693.384448, 216.66225228352)</SpatialMask>
 <Report alias="Points of interest" name="Poi">
 <Header>
   <Item>SCHOOL</Item>
   <Item>STREET</Item>
   <Item>HN</Item>
   <Item>ZIPCODE_N</Item>
   <Item>ZIPCODE_C</Item>
 </Header>
 <Records>
  <Record key="4" layer="lyr_PubSites">
    <Item>Peuterspeelzaal bezige bij</Item>
    <Item>CURIESTREET</Item>
    <Item>5</Item>
    <Item>7555</Item>
    <Item>NA</Item>
   </Record>
 </Records>
 </Report>

 <Report alias="Zoningplan" name="zoning">
 <Header>
   <Item alias="Object ID">OBJECTID</Item>
 </Header>
 <Records>
 <Record key="2" layer="hengelo_ora">
   <Item>2</Item>
 </Record>
 <Record key="128" layer="hengelo_ora">
   <Item>128</Item>
  </Record>
  <Record key="129" layer="hengelo_ora">
    <Item>129</Item>
  </Record>
 </Records>
 </Report>

 <MapResolution>2.224628</MapResolution>
 <XCoord>249519.960500</XCoord>
 <YCoord>476575.223000</YCoord>
 </FMSendInfo>
```

3.10.1 User keyinfo as a spatial search criterium

As explained before you can request attribute information based on coordinates or spatial masks that you add in the request. It is also possible to use keyinfo (graphical key's of geometries) as the input mask for a FMGetInfo request.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<FMGetInfo
  version="2.1.0"
  xmlns="http://www.bentley.com/FlexiMap"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bentley.com/FlexiMap FMRequest.xsd">

  <Service>demo_layers</Service>
  <KeyInfo displayMargin="0.1">
       <Keys layer="hoofdbestemmingen" keys="94"/>
  </KeyInfo>
  <MapResolution>0.5</MapResolution>
  <MapWidth>600</MapWidth>
  <MapHeight>500</MapHeight>
</FMGetInfo>
```

Based on this request Geo Web Solutions will search the layers based on the provided keyinfo. When it finds the geometries, these will be used as the spatial mask for the attribute report.

3.10.2 User a bufferdistance with a FMGetInfo request

You can expand the search for attribute information by adding a bufferdistance to the FMGetInfo request. This will result in a buffer being created with the given distance around the clickpoint, spatial mask or geometries found based on the keyinfo (see paragraph 3.10.1) provided in the request You add the bufferdistance directy after the <ClickPoint>,<SpatialMask> or <KeyInfo> element:

```
<BufferDist>25</BufferDist>
```

3.11 The FMGetTooltip request

The FMGetTooltip request is very similar to the FMGetInfo request. It also returns attribute information based on coordinate information. However there are differences:

- Only accepts a clickpoint (X/Y coordinate) as input <ClickPoint>;
- FMGetTooltip only returns one columnvalue per report;
- Designed for usage in the Geo Web Solutions Flash application (showing tooltips when hovering over mapobjects).

An example of an FMGetTooltip XML request is:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<FMGetTooltip
  version="2.1.0"
  xmlns="http://www.bentley.com/FlexiMap"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bentley.com/FlexiMap FMRequest.xsd">

  <Service>serv1</Service>
  <ServiceKey>7972500262309834033</ServiceKey> <!-- optional element -->
  <ClickPoint>249323 476652</ClickPoint>
  <MapResolution>0.5</MapResolution>
  <ReportName>Zoningobjects</ReportName> <!-- optional element -->

</FMGetTooltip>
```

The response to this request will have the following structure:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<FMSendTooltip
  xmlns="http://www.bentley.com/FlexiMap"
  version="2.1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bentley.com/FlexiMap FMSendTooltip.xsd">

  <Service>serv1</Service>
  <TooltipItems>
    <Item alias="bplan_alias">Housing</Item>
    <Item alias="bplan_alias">Traffic</Item>
  </TooltipItems>
</FMSendTooltip>
```

3.12 The FMGetReport request

The FMGetReport request is a combination of the FMGetList and the FMGetInfo request. Based on the provided attribute values, the keys will be determined corresponding to the attribute values. These keys are used to create an attribute report FMSendReport (similar to FMSendInfo).
The reports that will be searched for attribute information must be defined in de searchquery definition in the configuration file. Here you can define more then one reports as follows:

Fig: example of configuration that allows an FMGetReport request to immediately return attribute information based on the report(s) in the searchquery defnition (bold section)

```
 <SearchQuery name="Adres" alias="Search based on adres info" type="key">
      <ReportName>Info</ReportName>
      <KeyColumn type="alphanumeric">ObjectID</KeyColumn>
      <Columns>
        <Column type="alphanumeric" label="OBJNAME" minChars="0">OBJNAME</Column>
      </Columns>
      <MapResolution>0.5</MapResolution>
        <ReportNames>
            <ReportName>infobasic</ReportName>
            <ReportName>infodetail</ReportName>
        </ReportNames>
     </SearchQuery>
```

If you do not add a <ReportNames> section in the search configuration then the SendReport will be based on the report used to execute the search (in the above example the report Info).

Note: the ***keycolumn*** used in the attribute search needs to be the same as the keycolumn in the specified report(s).

If more then one reportnames have been added to the searchquery definition then you can specify which reports you want to see by adding the <ReportName> element.

Here is an example of an FMGetReport request:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<FMGetReport
  version="2.1.0"
  xmlns="http://www.bentley.com/FlexiMap"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bentley.com/FlexiMap FMRequest.xsd">

  <Service>serv1</Service>
  <ServiceKey>7972500262309834033</ServiceKey> <!-- optional element -->
  <SearchQuery name="mysearch1">
    <Column name="searchcol1" operator="like">STR%</Column>
  </SearchQuery>
  <ReportName>name of report </ReportName> <!-- optional element -->

</FMGetInfo>
```

The result of the FMGetReport request will have a structure as shown below:

- **header information**: name of each databasecolumn and alias if specified.
- A collection of **records** containing 1 or n records.
- Each record contains
    - the key of the geometry found (i.e. to be used to highlight geometries)
    - the layer in which it was found (i.e. to be used to highlight geometries)
    - values for all the databasecolumns defined in the header information

See also paragraph 3.10 (the FMGetInfo request).

3.13 The FMGetReportByKeyInfo request

The FMGetReportByKeyInfo request returns an attribute report (FMSendReport) based on key/layer information. Based on the provided layername and key(s) a report can be generated from all the reports that are linked to the specified layer.

Here is an example of an FMGetReportByKeyInfo request:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<FMGetReportByKeyInfo
  version="2.1.0"
  xmlns="http://www.bentley.com/FlexiMap"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bentley.com/FlexiMap FMRequest.xsd">

  <Service>serv1</Service>
  <ServiceKey>7972500262309834033</ServiceKey> <!-- optional element -->
  <KeyInfo>
       <Keys layer="mygeolayer" keys="104,6"/>
  </KeyInfo>
  <ReportName>name of report </ReportName> <!-- optional element -->

</FMGetReportByKeyInfo>
```

The result of the FMGetReportByKeynfo request will have a structure as shown below:

- **header information**: name of each databasecolumn and alias if specified.
- A collection of **records** containing 1 or n records.
- Each record contains
    - the key of the geometry found (i.e. to be used to highlight geometries)
    - the layer in which it was found (i.e. to be used to highlight geometries)
    - values for all the databasecolumns defined in the header information

See also paragraph 3.10 (the FMGetInfo request).

3.14 The FMGetReportByKeys request

The FMGetReportByKeys request is very similar to the FMGetReportByKeyInfo request. Here, the result is also an attribute report (FMSendReport). Only now it's solely based on keys. The <ReportName> element is mandatory in this request. The available report names are listed in the response to the FMGetFlashGUI request.

Here is an example of an FMGetReportByKeys request:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<FMGetReportByKeys version="2.1.0" xmlns="http://www.bentley.com/FlexiMap">
 <Service>demo_layers</Service>
 <ServiceKey>7972500262309834033</ServiceKey> <!-- optional element -->
  <Keys>104,6</Keys>
  <ReportName>MyReport</ReportName>
</FMGetReportByKeys>
```

The result of the FMGetReportByKeys request will have a structure as shown below:

- **header information**: name of each databasecolumn and alias if specified.
- A collection of **records** containing 1 or n records.
- Each record contains
    - the key of the geometry found (i.e. to be used to highlight geometries)
    - the layer in which it was found (i.e. to be used to highlight geometries)
    - values for all the databasecolumns defined in the header information

See also paragraph 3.10 (the FMGetInfo request).

3.15 The FMGetLocationByKeyInfo request

The FMGetLocationByKeyInfo request is very similar to the FMGetLocation request. Here, the result is also location information (FMSendLocation).

Here is an example of an FMGetLocationByKeyInfo request:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<FMGetLocationByKeyInfo version="2.1.0"
       xmlns="http://www.bentley.com/FlexiMap"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.bentley.com/FlexiMap FMRequest.xsd">

  <Service>myservice</Service>
  <ServiceKey>7972500262309834033</ServiceKey> <!-- optional element -->
  <KeyInfo>
    <Keys layer="mylayer" keys="336"/>
  </KeyInfo>
  <MapWidth>600</MapWidth>
  <MapHeight>550</MapHeight>

</FMGetLocationByKeyInfo>
```

The XML response to the FMGetLocationByKeyInfo request has the following structure:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<FMSendLocation
  xmlns="http://www.bentley.com/FlexiMap"
  version="2.1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bentley.com/FlexiMap FMSendLocation.xsd">

  <Service>myservice</Service>
  <MapResolution>0.500000</MapResolution>
  <XCoord>249301.100000</XCoord>
  <YCoord>476669.500000</YCoord>

</FMSendLocation>
```

The FMGetLocationByKeyInfo will return the centerpoint X and Y coordinates and Mapresolution that corresponds with the MBR of the geometries found based on the key information.
If no geometries can be found based on the provided keys the following response will be send to the client:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<FMServiceExceptionReport xmlns="http://www.bentley.com/FlexiMap" version="2.1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.bentley.com/FlexiMap FMServiceException.xsd">
  <ServiceException code="0x841FFFFF">
  <Reason>Handle GetLocationByKeyInfo request</Reason>
  <Description>No valid geometries are given to this request</Description>
</ServiceException>
</FMServiceExceptionReport>
```

3.16 The FMGetMapByKeyInfo request

The FMGetMapByKeyInfo request is very similar to the FMGetLocationMap request (see next paragraph). Here, the result is also a mapimage in jpeg or png format. Only now the location of the map is determined by a combination of layer and key(s).

Here is an example of an FMGetMapByKeyInfo request:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<FMGetMapByKeyInfo version="2.1.0" xmlns="http://www.bentley.com/FlexiMap"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.bentley.com/FlexiMap FMRequest.xsd">
 <Service>Demo_Layers</Service>
 <ServiceKey>7972500262309834033</ServiceKey> <!-- optional element -->
 <ContentType>image/png</ContentType>

 <KeyInfo displayMargin="0.2">
       <Keys layer="hoofdbestemmingen" keys="75"/>
 </KeyInfo>
 <HighlightResult>true</HighlightResult>
 <MapWidth>750</MapWidth>
 <MapHeight>333</MapHeight>
 <MinResolution>2.500000</MinResolution>
</FMGetMapByKeyInfo>
```

FMGetMapByKeyInfo gives you the possibility to **highlight** the geometries that where found during this request.

3.17 The FMGetLocationMap request

The FMGetLocationMap request is a “combined” request. It is a combination of the FMGetLocation and the FMGetMap request. This is also the only request that can be executed in two ways:

1. as an URL to the Geo Web Solutions ISAPI DLL with some parameters;
2. with an XML structure like all the other requests.

Ad 1) Using the URL
If you want to create a website that is basically text based, but you want to enrich the site with a mapimage based on for example address information. You can use the GetLocationMap URL in a HTML <image> tag. If you use the GetLocationMap URL, there are two ways in which you can format the URL:

http://<servername>/scripts/fleximap.dll?version=2.1.0&request=getlocationmap&service=bplan&**sq=adres&columns=straat,huisnr&values=bantingstraat,16**&mapwidth=600&mapheight=550&contenttype=image/jpeg &highlight=true

http://<servername>/scripts/fleximap.dll?version=2.1.0&request=getlocationmap&service=bplan&**sq=adres&straat=bantingstraat&huisnr=16**&mapwidth=600&mapheight=550&contenttype=image/jpeg &highlight=true

By default this url request returns a map image in the PNG file format that will have the defined Width and Height in pixels and will be zoomed to the extent corresponding to map objects found executing the search.

If you prefer an image in the JPEG file format you need to add the following optional parameter: **&contenttype=image/jpeg**.

Ad 2) Using the XML structure
If you prefer to use this *combination* request the same as all the other requests. This is the request:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<FMGetLocationMap
  version="2.1.0"
  xmlns="http://www.bentley.com/FlexiMap"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bentley.com/FlexiMap FMRequest.xsd">

  <Service>serv1</Service>
  <SearchQuery name="Adres">
    <Column name="STRAAT">CURIESTRAAT</Column>
    <Column name="HUISNR">5</Column>
  </SearchQuery>
  <ContentType>image/jpeg</ContentType>
  <HighlightResult>true</HighlightResult> <!-- optional element -->
  <MapWidth>304</MapWidth>
  <MapHeight>346</MapHeight>
</FMGetLocationMap>
```

Highlighting results:
The FMGetLocationMap gives you the possibility to highlight the geometries that where found during this request. The following options are available:

1. *You are performing the FMGetLocationMap request on a search that is based on XY coordinate value pairs*: You can show a map where the points are highlighted using a pointsymbol that is defined in the configuration.
2. *You are performing the FMGetLocationMap request on a search that is based on geometries stored in BGS- or OracleSpatial format*: You can show a map where the geometries are highlighted with a color and style that is defined in the configuration.

If you want to use this functionality, simply add the parameter/element as shown in the above examples in red.

4 Using the Geo Web Solutions Flash application

4.1 Introduction

One of the ways of creating a webapplication with Geo Web Solutions functionality is by using the flash client application. This flash application will communicate with the Geo Web Solutions server via XML and provides a lot of clientside "*out of the box*" functionality like zoom, pan and select functions. Advanced functionality like redlining, spatial queries and attribute searches are also available in the Geo Web Solutions Flash client.
In the configuration for a service you can define the available functionality in the flash client. Also you can define the *look and feel* of the flash client. This means that the flash client can be fully integrated in any website or workflow.
This chapter describes the ways in which the Geo Web Solutions Flash client can be integrated and used in a website. After reading this chapter you will be able to create a webpage that embeds the flash client and communicate with this flash application and "*catch"* the parameters that are exposed by Flash to the webpage.

Note:
The Geo Web Solutions flash application will only work if the following version of the Flash player is present on clientcomputers:
The minimal version of the Flash player on a client machine is version 7.0.19
If this version is not present on a client computers an error message appears, and the application will <u>not</u> work.

4.2 Embedding flash in a webpage

The Geo Web Solutions flash client consists of two swf files (**fmBase.swf** and **fmClient.swf**) and some asset swf files with icons that are used throughout the application.

The **fmBase.swf** and the asset files will always be loaded from the Geo Web Solutions server. They need to exist in a (virtual) directory directly beneath the root of the web server (see Chapter 2 for more information on directory structure).
The **fmClient.swf** is the file that will actually be *embedded* in a webpage. It is this webpage that is mainly responsible for the way the Geo Web Solutions Flash application will work and which "services" it will use. The fmClient.swf will usually be loaded from the webserver that serves the website that integrates the flash application.

The following information is important if you are embedding the Flash object in your webpage:

- ***The start parameters for the fmClient***: a defined list of parameters that allow the fmClient to instantiate and call the desired Geo Web Solutions service.
- ***Width and height of the Flash object in pixels***: the minimal width and height of the Flash object is **260x260** pixels. If the Flash object is embedded with smaller values a "*Format error*" will be shown and the application will <u>not</u> start. If the requested *service* has searchqueries defined in its configuration and if the settings allow the flash application to show the search window, the minimal height of the flash object should be 400 pixels. If this is not the case, flash will start but the search functionality will not be available.

Note: in internet explorer it is possible to embed the flashmovie using relative width and height values (percentages). However thisis not recommended, because this can result in strange behaviour in the flash movie. In a mozilla based browser the flash movie will not function with relative width and height values.

- ***Flash player version check***: the webpage should ideally do a version check of the installed Flash player on the client computer. If the installed Flash player has an older version number than version 7.0.19, the webpage should automatically start downloading the correct version of the Flash player.

The setup of Geo Web Solutions contains a demo website/sdk with an example that embeds the fmClient.swf in a HTML page. Here's an example of the HTML tags needed to embed the fmClient.swf:

```
<object
  classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"

codebase=http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=7,0,1,9
width="400" height="400" id="fmClient" align="middle">

  <param name="allowScriptAccess" value="sameDomain" />
  <param name="movie"
value="fmClient.swf?fm=http://myserver.com/scripts/fleximap.dll&service=bplan&servicegroup=BP&
commurl=http://myserver.com/fleximap/test/report.asp&commtarget=toonReports&sq=Adres&Straat=BO
ERHAAVELAAN&Huisnr=112" />
  <param name="quality" value="high" />
  <param name="scale" value="noscale" />
  <param name="bgcolor" value="#ffff" />
  <param name=wmode value='transparent'>
  <embed src="fmClient.swf?fm=http://myserver.com
/scripts/fleximap.dll&service=bplan&servicegroup=BP&commurl=http://myserver.com
/fleximap/test/report.asp&commtarget=toonReport&sq=Adres&Straat=BOERHAAVELAAN&Huisnr=112"
quality="high" scale="noscale" bgcolor="#ffffff" width="400" height="400" name="fmClient"
align="middle" allowScriptAccess="sameDomain" type="application/x-shockwave-flash"
pluginspage="http://www.macromedia.com/go/getflashplayer" />

</object>
```

Note:
The syntax for embedding flash movies is different in MS Internet Explorer (IE) and Mozilla based browsers like Firefox. In IE a parameter with the name "movie" is present. The value is the name of the SWF file to load and the start parameters needed by the Geo Web Solutions flash application:
<param name="movie" value="fmClient.swf?....... />
In firefox an <embed> tag needs to be included with a SRC attribute containing the SWF filename and its parameters:
<embed src="fmClient.swf?....... />
Make sure you use both when you embed the Geo Web Solutions flash application in your web pages for maximum compliancy with webbrowsers used.

4.2.1 Startparameters for the Geo Web Solutions Flash application

The flash application needs some information in order to instantiate itself correctly. This information is provided by adding a list of startparameters to the fmClient.swf. Some of the parameters are obligatory, while others are optional.
Here is an example of the list of parameters that can be added to fmClient.swf (between brackets are optional parameters):

fmClient.swf?**fm**=www.bentley.com/fleximap.dll&**service**=demo
[&**servicegroup**=zoningplans&**servicekey**=max19NumericCharacters
&**commurl**=www.example.com/receiveReport.asp&**commtarget**=fraReports
&**startlayers**=parcels|streets
&**idxlayers**=districts|overview
&**highlight**=true
&**legend**=true
&**info**=false
&**layers**=parcels|zoningplan&**keys**=91,92,93|132
&**xcoord**=249123&**ycoord**=471622&**mapresolution**=1.5
&**sq**=zipcode&**postc**=1000%20CC&**nr**=15**]**

Explanation of the available startup parameters:

Name:	***Required:***	***Explanation:***
fm	Yes	URL to the Geo Web Solutions ISAPI DLL
service	Yes	The service that will be initialized by flash
servicegroup	No	The name of the group (of services) that will be used in flash to list

		the services that can be turned on/off in flash. If the service defined in the previous parameter has enabled the tool "services", all the services linked to the service group will be listed in a window in flash. Users can switch between services in the same flash instance. The active service will be used for all GetMap, GetInfo and GetTooltip requests from flash.
servicekey	No	Key that identifies this session. This key must be requested from the Geo Web Solutions server by the website that contains the flash application. The **FMGetServiceKey** XML request is used for this purpose. This parameter is only required if a service is secured using the service key functionality. Note: only pages coming from a predefined ip-address can request a servicekey from the Geo Web Solutions server.
commurl	No	URL that defines the page that will parse the XML response with report information (FMSendInfo). This parameter is only required if showing attribute information about mapobjects is desired. If flash is not using the *info button* this startparameter can be omitted (always used in combination with **commtarget**)
commtarget	No	Name of the frame where flash will post the XML response with attribute information (FMSendInfo). This parameter is only required if showing attribute information about mapobjects is desired. If Flash is not using the *info button* this startparameter can be omitted (always used in combination with **commurl**)
startlayers	No	If the service that will be used is configured with layers that can be turned on/off (WMS mode), the possibility exists to define which layers will be turned on at startup. This overrules the startlayers as definied in the configuration file (seperated with a 'pipe' character |).
idxlayers	No	If the service that will be used is configured with layers that can be turned on/off, the possibility exists to define which layers will be used to create the **indexmap**. It is good practice to define layers without much detail for the indexmap (seperated with a 'pipe' character |)
highlight	No	Parameter that determines if highlighting is enabled. Possible values are **true** or **false**. If set to true the result of a searchquery will be highlighted in the map.
legend	No	Parameter that determines if the legend window needs to be shown in flash (to turn on/off layers in the map). Possible values are **true** or **false** (default value is true).
info	No	Parameter that determines if the flash client application will create a GetInfo request if a user is in "information" mode. If set to false only the coordinates of the selection mask are passed to the client (via AddParams/Submitparams, see paragraph 4.3). Possible values are **true** or **false** (default value is true).
layers	No	Parameter that contains the names of the layers (seperated with a 'pipe' character |) in which objects can be found that need to be highlighted. Always used in combination with **keys** parameter. Example: &layers=parcels|zoningplan
keys	No	Parameter that contains the keys of the objects that need to be highlighted. Keys to highlight in one layer seperated by komma. To switch between layer use the 'pipe' character). Always used in combination with **layers** parameter. Example: &keys=90,91,93|132
xcoord 1)	No	X-coordinate of the centerpoint used at startup (always used in combination with **ycoord** and **mapresolution**)
ycoord 1)	No	Y-coordinate of the centerpoint used at startup (always used in combination with **xcoord** and **mapresolution**)
mapresolution 1)	No	Mapresolution used for generating the initial map at a certain scale (always used in combination with **xcoord** and **ycoord**)

sq [1)]	No	Name of the searchquery that will be executed after startup. If this parameter is specified the subsequent parameters should contain the values that the search will use
legend	No	Parameter that determines if the legend is enabled in flash. Possible values are **true** or **false**. If set to true the legend will be shown. If set to false the layers will be managed with the Flash API functions.
info	No	Parameter that determines if the Flash Client will create the GetInfo request. Possible values are **true** or **false**. If set to true then the FMGetInfo request will be generated by Flash. If set to false only the infocoordinates are exposed and the website will manage the getInfo requests.

1) If sq and column parameters are used but also xcoord, ycoord and mapresolution are present, then xcoord, ycoord, and mapresolution gets priority and will be executed.

Note: before the HTML page with the *embedded* flash movie is interpreted by the webbrowser all the startparameters that Geo Web Solutions needs should be present in the website (flash will only process the startup parameters when the page is being parsed by the webbrowser).

Note: An URL needs to consist of ‘safe characters’. This means that potentially ‘unsafe’ characters need to be URL encoded. For more reading on safe characters and encoding see RFC 2396.

4.3 Communication between flash and embedding webpage

The Geo Web Solutions flash application can “post” valuable information to the webpage that contains the fmClient.swf. This information can be used to:

- Store the values that the map is composed of in a database;
- Recreate the map that is currently shown in the flash environment;
- Store all the coordinate information of redlined objects;
- Recreate the same map with all the redlined information (as a JPEG or PNG image);
- Create a GetInfo request based on the spatial mask that is drawn in Flash (in ‘info’ mode)
- Et cetera.

Flash will “submit” the information if the currently active service has the **UseJavascript** property set to **true**. If this is the case, flash expects these two JavaScript functions to be defined in the webpage:

```
// PUBLIC VARIABLE TO STORE THE EXPOSED STRING
var exposedParams;
```

```
function addParams(params)
{
        // DO NOT PUT ALERT BOXES IN THIS FUNCTION, IT IS CALLED WITH AN INTERVAL
        exposedParams += params;
}

function submitParams()
{
        // THIS FUNCTION IS CALLED WHEN THE ENTIRE STRING HAS BEEN EXPOSED
        // HERE YOU CAN EXEWCUTE YOUR OWN CODE TO…
}
```

The function addParams(params) accepts 1 argument: **params**. In this argument Flash will pass a ‘(sub)string’. The function will be called in a loop until the entire string with information has been exposed. At that time the function submitParams() is called to inform the webpage the entire string has been exposed.
The result is a long string separated by pipes “|” that contains all the parameters and it’s corresponding values. The

The JavaScript function submitParams() { } can be interpreted as an eventhandler that will be executed each time a:

1. new mapimage is generated in Flash;
2. information is requested in Flash.
3. objects are redlined on the map in Flash

Ad (1)
The following string with parameters will be passed to the JavaScript functions every time flash has requested a new mapimage:

fm=http://mywebserver/scripts/fleximap.dll|service=serv1|servicekey=123456789|mapwidth=400|mapheight=314|mapresolution=3|xcoord=249504.379066|ycoord=476683.194699|maplayers=zoningplan,parcels

The parameters indicated in **black** will always be exposed after flash requests a mapimage. The parameters in **red** will only be exposed if necessary:
servicekey: only if the current service is secured
maplayers: only if the service is configured to work with layers that can be turned on/off

This string will be posted to the argument of the JavaScript addParams function in 'chunks'. When the entire string is exposed to this function, submitParams is called. From here on it's up to the developer of the website what will be done with the exposed string. In the flashtest site that is delivered with Geo Web Solutions this string is split into an array and each parameter is stored separately for later use.

The parameters exposed provide you with all the information you need to generate the same maprequest as the one executed in flash. This can be used for example to create a printpage.

Ad (2)
If a user is in "information mode" in flash and uses one of the selection methods (point, line, circle or polygon) to select objects on the map, a request for information is send to the Geo Web Solutions server (FMGetInfo request). When this happens the following string with information is posted to the JavaScript function:

Selecting using a point:
Info=POINT(249408.379066 476689.194699)
(coordinates of point clicked)

Selecting using a line:
Info=LINESTRING(249405.749987 476720.916673,249649.833354 476452.666636)
(coordinates defining the line)

Selecting using a circle:
Info=CIRCLE(249483.083331 476969.833374, 99.6417330294137)
(center point of circle and radius)

Selecting using a polygon:
Info=POLYGON((249606.333348 476730.583341,249427.49999 476476.833306,249599.083347 476457.49997,249676.416691 476720.916673,249606.333348 476730.583341))
(coordinates defining the polygon, first and last point are the same)
Note: Beware the double brackets around the coordinates at the Polygon example. The reason for this is that the geometry definition has been based on the Well-known Text definition for geometric objects.

Ad (3)
If a user is in "redline mode" in flash the following "drawing tools" are available to redline geometries: point, line, circle, polygon or text. Each time the user finishes drawing a redline geometry (after a doubleclick) on the map, the following string is posted to the JavaScript function:

Redlining point geometry:
Redline=POINT(249408.379066 476689.194699)
(coordinates of point clicked)

Redlining line geometry:
Redline =LINESTRING(249405.749987 476720.916673,249649.833354 476452.666636)
(coordinates defining the line)

Redlining circle geometry:
Redline=CIRCLE(249483.083331 476969.833374, 99.6417330294137)
(center point of circle and radius)

Redlining polygon geometry:
Redline=POLYGON((249606.333348 476730.583341,249427.49999 476476.833306,249599.083347 476457.49997,249676.416691 476720.916673,249606.333348 476730.583341))
(coordinates defining the polygon, first and last point are the same)

Redlining text:
Redline=TEXT(249052.916605 477136.583397, "This is text")
(coordinates where text starts and the text in quotes)
Note: the text is posted "as is" to the JavaScript function. This means that some characters (like a single quote) are not allowed in the string. This causes a JavaScript error on the webpage.

Clearing redlined objects:
Redline=CLEAR
(If a user deletes all the redline objects in the flash application, this string is posted)

Table: Explanation of parameters that will be posted to JavaScript function addParams:

ParamName:	*Explanation:*	*When posted?*
fm	URL to the Geo Web Solutions server (fleximap.dll)	Each time a new mapimage is requested in flash (after a GetMap or GetLocationMap request)
service	Name of the service that is currently active in flash and was used to create the current mapimage.	Each time a new mapimage is requested in flash (after a GetMap or GetLocationMap request)
servicekey	Encrypted key that is valid at the current moment for this service on the Geo Web Solutions server	Each time a new mapimage is requested in flash (after a GetMap or GetLocationMap request) **but** only if the service is secured with a servicekey. If this is the case the website needs to obtain a valid servicekey with the GetServiceKey request.
mapwidth	Width of the current mapimage in pixels	Each time a new mapimage is requested in flash (after a GetMap or GetLocationMap request)
mapheight	Height of the current mapimage in pixels	Each time a new mapimage is requested in flash (after a GetMap or GetLocationMap request)
mapresolution	Number of 'map units' per pixel (this defines the scale) of the current mapimage.	Each time a new mapimage is requested in flash (after a GetMap or GetLocationMap request)
xcoord	X-coordinate of the centerpoint of the map	Each time a new mapimage is requested in flash (after a GetMap or GetLocationMap request)
ycoord	Y-coordinate of the centerpoint of the map	Each time a new mapimage is requested in flash (after a GetMap or GetLocationMap request)
Info	Coordinate information of the clickpoint or spatialmask used as	Each time the user is in "Information" mode and uses a point, line, circle or polygon to request attribute information

	input for an information request	
Redline	Coordinate information of the geometries that have been redlined on the map	Each time the user is in “Redline” mode and draws a point, line, circle, polygon or text on the map. Information of each object is posted after completing the redlined object (double click for all object except point)

As mentioned before it’s up to the developer how this information will be processed. It can be used to reproduce the mapimage, to store the mapimage in a database or send it by email. The redlined information can also be used to store this information in a (spatial) database. All the information is available for creating your own FMGetMap or FMGetInfo requests.

Example of the JavaScript functions to ‘catch’ the parameters:

```
/*-------------------------------------------------------------*
 * addParams:                                                  *
 * This function will be used by Flash to expose parameters   *
 * Function must be named "addParams" and accept 1 parameter  *
 * ==============Content must NOT be changed================= *
 * If string from Flash is too long, this function will be    *
 * called until entire string is passed                       *
 *-------------------------------------------------------------*/
function addParams(params)
{
       // DO NOT PUT ALERT BOXES IN THIS FUNCTION
       // FUNCTION IS CALLED WITH AN INTERVAL
       exposedParams += params;
}
/*-------------------------------------------------------------*
 * submitParams:                                               *
 * This function will be called from Flash when the entire    *
 * string is transferred to addParams                         *
 * Now a function can be called to process the exposed params *
 *-------------------------------------------------------------*/
function submitParams()
{
       // NOW YOU CAN CALL YOUR FUNCTIONS TO DEAL WITH THE PROVIDED INFO
       // OR YOU CAN PROCESS THEN IN THIS FUNCTION

       myownfunction(exposedParams);
}

function myownfunction(params)
{
    var arrAllParams = params.split('|');
    for (var i = 0; I < arrAllParams.length; i++)
    {
        var tmp = arrAllParams[i];
        var arrOneParam = arrAllParams[i].split('=');
        var param = arrOneParam[0].toLowerCase();
        var val= arrOneParam[1];
        switch (param)
        {
            // Do your thing with the provided parameters in the array
        }
    }
    return;
}
```

4.4 JavaScript API for communication with the flash client

It is possible to communicate with the flash client application using javascript functions. This is realized by creating a Gateway that allows the communication.

The following API functions have been defined:

Table: API functions:

API function name:	***Parameters:***	***Explanation***
I_getMapByKeyInfo	sellayers:String keys:String displayMargin:Number	This function will center the map around the elements that can be found based on the keys provided. Also the elements will be highlighted. sellayers can contain 1 or more layernames. The layers are separated by the "|" character. The keys parameter will contain the keys comma-separated for a specific layer. displayMargin is an optional parameter that can contain a value from 0 to 0.5. *Example: I_getMapByKeyInfo("layer1|layer2", "10,20,30|2,3",0.2)*
I_GetLocationMap	sqName:String colStr:String valStr:String	This function will execute the search and will shouw the resulting mapimage. sqName is the name of the searchquery to execute; colStr is a comma separated string with fieldnames; valStr is a comma separated string with the values.
I_DrawRedline	wkRedline:String	This function will draw the coordinates provided in the parameter wkredline on the map. The wkRedline parameter has the same syntax as exposed by flash when object are redlined on the map (see paragraph 4.3).
I_ClearRedline	n/a	This function will clear the redline layer in flash
I_IndexMap	action:String	This function will open or close the indexmap. Possible values for parameter action are **open** and **close**.
I_ClearSelection	n/a	Calling this function will cause the map to be refreshed, herebye clearing the selection.
I_UpdateMap	vlayers:String alayers:String	Set the layers that should be <u>visible</u> (vlayers) and the layers that should be <u>active</u> (alayers). Calling this function will result in updating the map.
I_SetActiveTool	toolname:String	Change the currently active toolbar function using the API. Possible values are: **zoomin, zoomout, info, redline, pan**.
I_setCommParams	c_url:String c_target:String	Change the url that will be executed when a GetInfo request is being posted by flash. Also the target can be changed. If one of the values is not definied or an empty string then the values as defined during startup (see paragraph 4.2.1)will be used.
I_mapNewCenter	X:String Y:String mapRes:String	function to recenter and/or rescale the map in flash based on x and y coordinates and a (optional) mapresolution determining the scale

The API uses a swf file (fmgateway.swf) that needs to loaded silently in the background to communicate with the Geo Web Solutions flash application. How this can be implemented is explained in the APItest pages. See paragraph 5.2.

5 Flash client test website for Geo Web Solutions

Note that the Flash client testpage scripts are being distributed with Geo Web Solutions "as is". Their purpose is to:

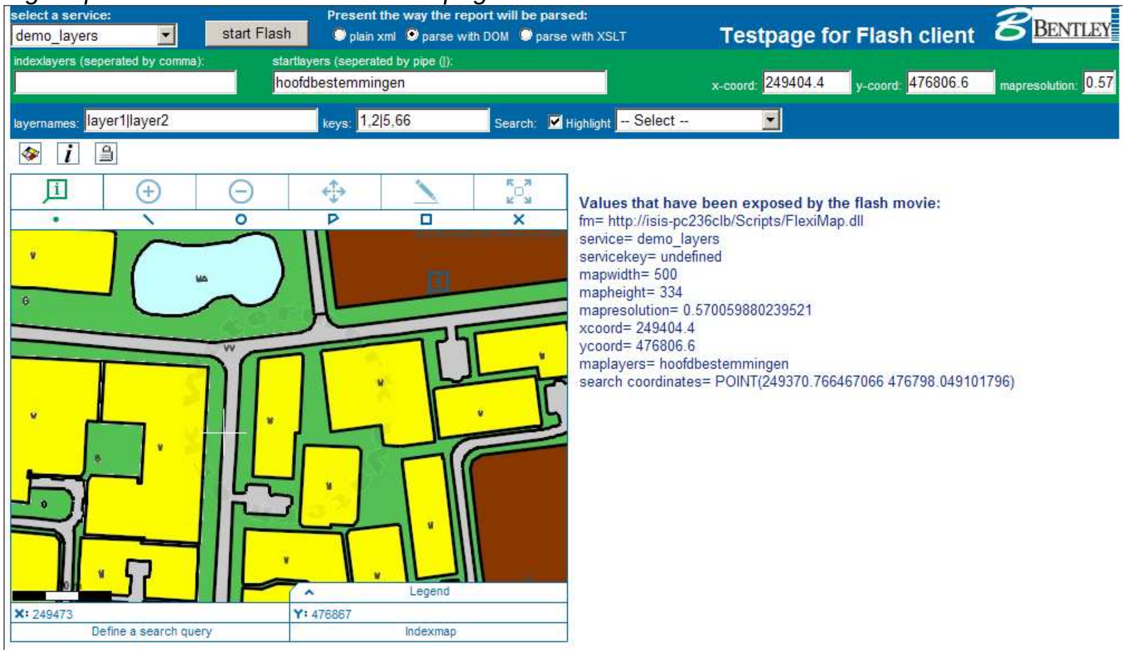
1) test if GWS and/or the flash client is functioning properly.
2) provide code examples for webdevelopers

No rights can be derived from these scripts.

This chapter describes the structure of the flashtest website that is delivered with Geo Web Solutions. This website has been designed to demonstrate almost all the possible ways in which you can use the components that are provided by Geo Web Solutions. By looking at this code you can get a good idea of how to integrate Geo Web Solutions components in your websites.

5.1 Flash test page

Fig: Impression of the flash client testpage



select a service: demo_layers

Present the way the report will be parsed: plain xml [✓ parse with DOM] parse with XSLT

indexlayers (seperated by comma):

startlayers (seperated by pipe (|)): hoofdbestemmingen

x-coord: 249404.4 y-coord: 476806.6 mapresolution: 0.57

layernames: layer1|layer2 keys: 1,2|5,66 Search: ✓ Highlight -- Select --

Values that have been exposed by the flash movie:
fm= http://isis-pc236clb/Scripts/FlexiMap.dll
service= demo_layers
servicekey= undefined
mapwidth= 500
mapheight= 334
mapresolution= 0.570059880239521
xcoord= 249404.4
ycoord= 476806.6
maplayers= hoofdbestemmingen
search coordinates= POINT(249370.766467066 476798.049101796)

5.1.1 Files used in the flashtest site

If you performed a default installation of Geo Web Solutions then you can find the flashtest page in "c:\inetpub\wwwroot\geowebsolutions\flashtest". Here you will find the following scripts.

Table: description of the scriptfiles used in the flash client testpage

File name	Description
demo.js	Contains all the JavaScript functions used in this website
FlashAPIfuncs.js	Contains the JavaScript functions for the API testsite (see next paragraph)
index.html	Startpage of the flash testsite.
indexAPI.html	Startpage of the flash API testsite.
Info.html	An empty container html file
report.asp	Serverside asp script that catches and processes (parses) the response to a FMGetInfo request.

report.xsl	xslt file used to transform the XML response (FMSendInfo) into HTML to show the attribute information
report2.asp	Asp script to parse FMGetInfo request
Reportblank.asp	Asp script to parse FMGetInfo request
Reportplain.asp	Asp script to parse FMGetInfo request
servicekey.asp	Serverside asp script that will create a FMGetServiceKey request to obtain a valid key to use in a secured service
Styles.css	Style sheet used in the demo (colours, font size, etc.)

The easiest way to familiarise yourself with the different components that Geo Web Solutions has to offer is by examining this code.

This demosite provides examples of:

- How to generate valid XML requests using JavaScript
- How to start and use the flash client
- How to “catch” and “process” the parameters exposed by flash
- How to parse to xml response with attribute information
- How to request and use servicekeys

5.1.2 Starting the flash test page

When you start index.html the website get a list of all available services for the current service by executing a FMGetServiceList request from a Javascript function (**getServices**) and showing the response in a dropdownlist.

If you select a service a JavaScript function is triggered that will retrieve a list of all searches defined for the selected service (**getAvailSearches**). On selecting a search the questions related to this search will be executed (**getQuestion**) starting with the first question and onwards.

After all the questions have been answered the flash client will be embedded in the html with the correct startparameters (**buildStartParams** and **startFlash**).

Looking at these JavaScript functions in *demo.js* will give you a good idea of how to create valid xml requests to the Geo Web Solutions server and process the response in a regular HTML page.

5.1.3 Startparameters for the flash client

At the top of the flash testpage there are some textfields present that can be used to enter startparameters that can be used to determine the behaviour of the flash client after startup.
You can enter coordinate (XY) information and mapresolution, startlayers, layernames and keys to highlight et cetera.

When all the startparameters have been provided, press the start button at the top of the page to embed the flash client with the right parameters.

5.1.4 Catch and process exposed parameters

Everytime when flash exposes information to the Javascript functions addParams(param) and submitParams(), the exposed string with parameters will be processed in the function submitParams().
This function will break up the string and store each parameter in public variables.

Two buttons at the top of the page will do something with the exposed information.

1) One will create a FMGetMap request based on the exposed paramters and will show the resulting map.
2) Clicking on the "i" button will show the parameters that have been exposed.

5.1.5 Parse xml response with attribute information

When the user is in info mode, flash composes and submits FMGetInfo requests. The response to such a request is a XML response (FMSendInfo) that flash will immediately "pass-on" to a defined page at a specific location. Which page and what location is defined in 2 startparameters for flash: commurl (script to submit the xml to) and commtarget (frame to execute this script). The XML response is posted with the HTTP POST method.
In the flash Client testpage, the xml response is parsed by Report.asp. At the top of the page you can select how the response will be parsed:

- Plain xml
- Parsing with DOM
- Parse with XSLT

As you can see the same report can be visualized as you see fit. It allows you to fully integrate attribute reports in you website.

5.1.6 Integrating servicekeys in your website

A Geo Web Solutions service can be secured with a servicekey. If this is the case then almost every request needs to contain a valid servicekey. These keys can be obtained only from computers with an ip-address known to the Geo Web Solutions server.
a servicekey has a (configurable) expiration time. If you are working with secure services you need to keep this in mind.
The flash client testpage contains functionality that can be used as an example of how to implement servicekeys. If a service is secured, the website will note this. The user can then press the "lock" button. This will start a serverside script *servicekey.asp* that will request a valid servicekey from the Geo Web Solutions server. It can do so because the ip-adress of the machine running *servicekey.asp* is defined in the GWS configuration.

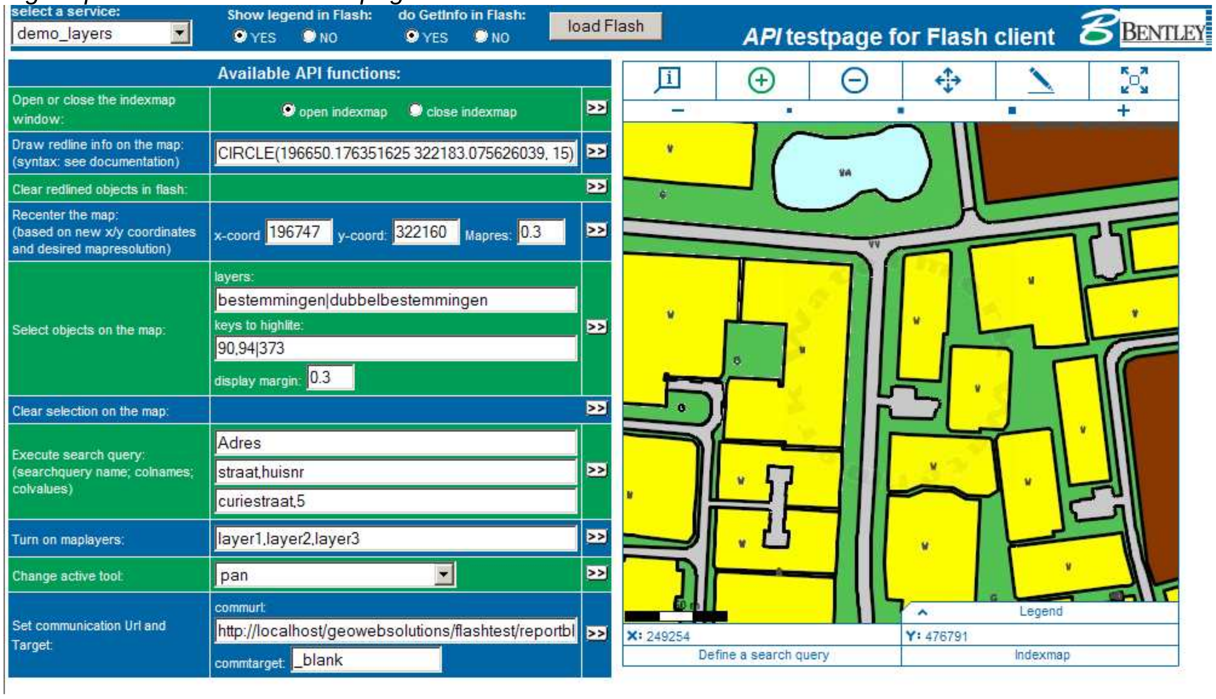
5.2 Using the javascript API for communication with the flash Client

When you use indexAPI.html (or when you click on the Bentleylogo in the Flash testpage) then you will get another testpage. This page allows you to examine how the API works. You can use the API to communicate with Flash without restarting Flash.

Table: additional files needed for the API to function properly

File name	Description
fmGateway.swf	flash movie that will load silently and will be used for communication with the GWS flash movie (default installed in c:\inetpub\wwwroot\geowebsolutions\fmGateway.swf)
fmGateway.js	Javascript page that contains code to set up the communication. (default installed in c:\inetpub\wwwroot\geowebsolutions\fmGateway.swf)

Fig: Impression of the API testpage



select a service: demo_layers | Show legend in Flash: [✓ YES] NO | do GetInfo in Flash: [✓ YES] NO | load Flash

API testpage for Flash client

Available API functions:		
Open or close the indexmap window:	[✓ open indexmap] close indexmap	>>
Draw redline info on the map: (syntax: see documentation)	CIRCLE(196650.176351625 322183.075626039, 15)	>>
Clear redlined objects in flash:		>>
Recenter the map: (based on new x/y coordinates and desired mapresolution)	x-coord 196747 y-coord: 322160 Mapres: 0.3	>>
Select objects on the map:	layers: bestemmingen|dubbelbestemmingen keys to highlite: 90,94|373 display margin: 0.3	>>
Clear selection on the map:		>>
Execute search query: (searchquery name; colnames; colvalues)	Adres straat,huisnr curiestraat,5	>>
Turn on maplayers:	layer1,layer2,layer3	>>
Change active tool:	pan	>>
Set communication Url and Target:	commurl: http://localhost/geowebsolutions/flashtest/reportbl commtarget: _blank	>>

These scripts provide a basic test for all the API functions that are supported in this version of Geo Web Solutions.

All the functions as described in paragraph 4.4 have been implemented in this testpage.
The idea behind creating an API is having the possibility to communicate with the flash client without constantly starting Flash again. Some examples:

1. If you start Flash with the startparameter **legend=false** then flash will not show the legend to turn on/off layers. You now must manage visible layers using the API (function *I_updateMap)* . This allows a developer to create the legend in html using the legendinformation that is retrieved using a FMGetFlashGUI/FMGetCapabilites request.
2. If you start Flash with the startparameter **info=false** then flash will not create a getinfo request (and pass it on to the script defined in the commURL parameter. This allows a developer to get the spatial mask info when a user clicks in flash in info mode and create attribute reports based on one of the available Info requests.
3. Flash can be provided with new search values without re-initializing (function *I_getLocationMap*).
4. Flash can be provided with new x/y coordinates without re-initializing (function *I_mapNewCenter*).
5. et cetera.

The API works via a gateway.swf file that will be loaded in the background. The only thing that you need to be aware of is that this gateway.swf must be loaded with a key. As a key we use the service name that is being used in the application at that time. You can check out how this works in:

In **flashAPIFunc.js** the communication swf is created (the uid provided is the name of the selected service:
flashProxy = new FlashProxy(_**service**, _flashGateway);

In c:\\inetpub\wwwroot\geowebsolutions\fmgateway.js this function is defined:

function FlashProxy(uid, proxySwfName)
{

```
    this.uid = uid;
    this.proxySwfName = proxySwfName;
    this.flashSerializer = new FlashSerializer(false);
}
```

6 Troubleshooting Geo Web Solutions

6.1 Introduction

This chapter gives an overview of possible errors and solutions

Possible log file level-settings:
0 = None;
1 = Error;
2 = Error + for example SQL queries;
3 = Warning;
4 = Warnings + handy log messages;
5 = Maximum log level.

Error	Description	Solution
XML DOM Parse error: Unterminated entity reference	This error can occur when an illegal character is used in the configuration file, for instance an ampersand (&).	Enter an ampersand like this **&#x26;** or **&amp;** If you use the Geo Web Solutions Administrator, special characters will be correctly encoded
XML DOM Parse error: Error at file , line xx, char xx Message: Element 'xx' is not valid for content model	Either an invalid element is present in the XML, OR the order of appearance is not compliant with the schema.	Change the order of the elements in the XML request. You can find more information about the available XML requests in this developer guide.
Using the flash application: a JavaScript error (object expected)	Every time a new map is requested, a user redlines objects or queries for information (info mode) a JavaScript error occurs.	Check if the webpage that embeds fmClient.swf contains a JavaScript function named: addParams(param) and submitParams(). If UseJavaScript is set to true for the active service then flash will try to post parameters to these functions. If it does not exist, this error occurs.
Search window does not appear in flash application	The configuration of the service has searches defined and the functionality is enabled in the flash GUI, but the window is not loaded at startup.	Check the height (in pixels) of the embedded flash object. The minimal height needs to be 400 pixels. If the height is anywhere between 260 and 400 pixels, flash will start but not load the search window.
After requesting a mapimage (FMGetMap) the image cannot be save to harddisk in the webbrowser	After a FMGetMap request the resulting image is shown in the browser. But if saved to disk the resulting image cannot be read.	This is normal behaviour if you used the POST HTTP method. Try using the GET method instead of POST if you want the user to be able to download the image